Segmentation and Targeting in Banking

Objective: Segment potential customers into meaningful groups such as students, seniors, high-profile customers, etc., to tailor marketing strategies effectively.

Key Segmentation Algorithms

1. **K-Means Clustering:**
   - **Description:** A popular and straightforward clustering algorithm that partitions customers into K distinct non-overlapping subgroups.
   - **Use Case:** Effective for identifying broad segments like students, seniors, high-profile customers.
   - **Steps:**
     1. Standardize data (mean=0, variance=1).
     2. Choose the number of clusters K.
     3. Assign each data point to the nearest centroid.
     4. Recompute centroids as the mean of all points in a cluster.
     5. Repeat steps 3-4 until convergence.
   - **Example:**

     ```
     from sklearn.preprocessing import StandardScaler
     from sklearn.cluster import KMeans

     scaler = StandardScaler()
     df_scaled = scaler.fit_transform(df)

     kmeans = KMeans(n_clusters=5, random_state=0).fit(df_
     scaled)
     df['segment'] = kmeans.labels_
     ```

2. **Hierarchical Clustering:**

- **Description:** Builds a tree of clusters (dendrogram) by progressively merging or splitting clusters.
- **Use Case:** Useful when the number of clusters is not known a priori and for visualizing the clustering process.
- **Steps:**
    1. Compute the distance matrix.
    2. Merge the closest pair of clusters.
    3. Repeat until all points are merged into a single cluster.
- **Example:**

```
from scipy.cluster.hierarchy import dendrogram, linka
ge
import matplotlib.pyplot as plt

linked = linkage(df_scaled, method='ward')
dendrogram(linked)
plt.show()
```

3. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**
    - **Description:** Identifies clusters based on the density of data points, allowing for the identification of arbitrarily shaped clusters and noise.
    - **Use Case:** Effective for discovering clusters in data with varying densities and for outlier detection.
    - **Steps:**
        1. Choose parameters `eps` (maximum distance between points in a cluster) and `min_samples` (minimum number of points to form a dense region).
        2. Classify points as core, border, or noise.
        3. Expand clusters from core points.
    - **Example:**

```
from sklearn.cluster import DBSCAN
```

```
dbscan = DBSCAN(eps=0.5, min_samples=5)
df['segment'] = dbscan.fit_predict(df_scaled)
```

Important Variables for Segmentation (Indian Banking Perspective)

1. **Demographic Information:**
    - **Age:** Important for segmenting customers into groups like students, working professionals, and retirees.
    - **Income Level:** Helps differentiate between high-profile customers and others.
    - **Location:** Urban vs. rural, regional differences in banking needs.
2. **Behavioral Data:**
    - **Transaction History:** Frequency, volume, and type of transactions.
    - **Product Usage:** Types of banking products used (savings accounts, credit cards, loans).
    - **Channel Preference:** Online banking, mobile app usage, branch visits.
3. **Socio-economic Indicators:**
    - **Occupation:** Differentiates between salaried employees, self-employed individuals, etc.
    - **Education Level:** Can influence financial product preferences and needs.
4. **Credit Information:**
    - **CIBIL Score:** Indicative of creditworthiness, important for segments focused on lending products.
    - **Loan History:** Existing loans, repayment behavior.
5. **Customer Interaction Data:**
    - **Customer Service Interactions:** Frequency and type of inquiries or complaints.
    - **Feedback and Surveys:** Sentiment analysis from customer feedback.

Detailed Example of Segmentation Implementation

Step-by-Step Implementation Using K-Means Clustering

1. **Data Preparation:**
   - Collect data from various sources (transaction history, demographic data, credit information).
   - Handle missing values and outliers.
2. **Feature Engineering:**
   - Create relevant features (e.g., average monthly transaction volume, total number of products held).
   - Standardize numerical features to have mean = 0 and variance = 1.
3. **K-Means Clustering Implementation:**

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Sample data preparation
data = {
    'age': [23, 45, 56, 67, 34],
    'income': [30000, 80000, 120000, 150000, 50000],
    'transaction_volume': [5000, 15000, 30000, 50000, 12
000],
    'products_held': [1, 3, 4, 5, 2],
    'cibil_score': [700, 750, 800, 650, 720]
}
df = pd.DataFrame(data)

# Standardize the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

# Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=0).fit(df_sca
led)
df['segment'] = kmeans.labels_
```

```
print(df)
```

4. **Interpretation and Actionable Insights:**
   - Analyze the characteristics of each segment.
   - Develop targeted marketing strategies for each segment (e.g., student loans for the student segment, investment products for high-income segment).

Variables for Targeting

1. **Student Segment:**
   - Age: Typically 18-25
   - Education: Enrolled in higher education
   - Transaction Patterns: Frequent small transactions, usage of educational loans
2. **Senior Segment:**
   - Age: Above 60
   - Income: Pensioners, fixed income
   - Transaction Patterns: Less frequent transactions, higher focus on savings and fixed deposits
3. **High-Profile Customers:**
   - Income: Above a certain threshold (e.g., INR 1,00,000/month)
   - Occupation: Senior executives, business owners
   - Credit Information: High CIBIL score, high credit limits

By focusing on these key segmentation techniques and relevant variables, banks can effectively segment their customer base and tailor their marketing campaigns to meet the specific needs of each group.